LAB 5: Looping Constructs

OBJECT
To explore the concept of looping in a C programming language

THEORY
The concept of looping provides us a way to execute a set of instructions more than once until a particular condition is true. In C, loop is constructed by three ways.
1. The for loop
2. The while loop
3. The do while loop

The for Statement
The *for* loop is appropriate when you know in advanced how many times the loop will be executed. Here you have a counter variable whose limits are define. The general form of the for statement is

for (initialization of counter; loop continuation condition; increment counter)
statements;

The initialization expression initializes the loop's control variable or counter (it is normally set to 0); loop continuation condition determines whether the loop counter is within its limits or not and finally the increment statement increments the counter variable after executing the *inner loop statements*.

```
/* for loop*/
 void main(void)
 {

 int counter; /*Declaring a counter variable*/
 int table=2; /*To print the table of */
 clrscr();
 for(counter =1;counter <=10;counter++)
 printf("\n%3d *%3d =%3d",table,counter,table*counter);
 getch();
 }
```

The while statement: The while is an entry-controlled loop statement.

```
 while ( test condition)
 {
 body of loop
 }
```

The test condition is evaluated and if the condition is true then, the body of the loop is executed. This process is repeated until the test-condition finally becomes false and the control is transferred out of the loop.

Example: The following program asks the user to enter a number and displays the multiplication table of the given number.

```
void main ( )
```

```
{
 int num, k;
 clrscr( );
 printf("Enter any number :");
 scanf( "%d", &num);
 k =1;
 while ( k < = 10)
 {
 printf ( " %d x %d = %d \n", num, k, num *k);
 k = k+1;
 }
 }
```

Note: the loop we have written has been controlled by counting the number of iterations. In above code, the variable k in used for remembering the number of iterations.

Loop control using uncertain values: The method using counters can be used only when you know the number of data items to be read or processed. There are many problems where the number of values to be processed is not known at the beginning. In such a case the sequence of values in the input is terminated by a special value, which itself will never be a part of the sequence. This special value is referred to as the sentinel value.

Example:

```
/* while loop*/
void main(void)
{
 char guess; /*Declaring a counter variable*/
 clrscr();
 printf("Enter a character from a to z:");
 guess=getche();
 while(guess!='e')
 {
 printf("\nRetry!");
 printf("\nEnter a character from a to z:");
 guess=getche();
 }
 printf("\nThats it!");
 getch();
}
```

The do while Statement

The *do while* repetition statement is similar to the while statement. In the while statement, the loop-continuation condition test occurs at the beginning of the loop before the body of the loop executes. The *do while* statement tests the loop-continuation condition after the loop body executes; therefore, the loop body always executes at least once.

```
do
{
 Statement;
```

}

while (condition);

This loop must be executed at least once because the condition is checked at the end. If the condition is following while is true the control is transferred to the beginning of the loop statement otherwise control is transferred to the statement following while statement. **Example:** The following program asks the user to enter an integer and adds the individual digits of this number.

```
void main ( )
{
 int number, lastdigit, sum;
 clrscr();
 printf ("Enter any positive integer : ");
  scanf("%d", &number);
 sum = 0;
 do
 {
        lastdigit = number % 10; /* Extract the right most digit*/
        sum = sum + lastdigit; /* add it to sum */
        number = number /10; /*Remove the rightmost digit */
 }
 while (number >0);
 printf ("The sum of digits is = %d\n", sum);
 getch();
 }
```

Common Programming Errors

- Using commas instead of semicolon in a ***for*** header is a syntax error.
- Placing a semicolon immediately to the right of the header makes the body of that for statement an empty statement.
- Forgetting one or both of the braces that delimit a compound statement.
- Placing a semicolon after the condition in a while statement.
- If a counter or total is not initialized, the result of your program will probably be incorrect. This is an example of a logical error.
- Do not compare floating point numbers.
- Attempting to use the increment or decrement operator on an expression other than a simple variable name is a syntax error, e.g. writing --(y+3);

EXERCISE:

Q#01: Write a program to display the following patterns.

Answer:

```
#include <stdio.h>

int main() {
   int i, j;

   // Loop to print the pattern
   for(i = 1; i <= 5; i++) {
      for(j = 1; j <= i; j++) {
         printf("*");
      }
      printf("\n");
   }

   return 0;
}
```

```
*
**
***
****
*****
```

Q#02: Write a program that asks the user to enter any number, and the program display its factorial.

Answer:

```
#include <stdio.h>

int main() {
   int num, i;
   unsigned long long factorial = 1;

   // Taking input
   printf("Enter a number: ");
   scanf("%d", &num);

   // Factorial calculation using loop
   for(i = 1; i <= num; i++) {
      factorial *= i;
   }

   // Display result
   printf("Factorial of %d is %llu\n", num, factorial);

   return 0;
}
```

Q#03: Write a program using for loop that print the ASCII values from A to Z.

Answer:

```
#include <stdio.h>


int main() {
    char ch;

    // Loop through A to Z and print ASCII values
    for(ch = 'A'; ch <= 'Z'; ch++) {
        printf("ASCII value of %c = %d\n", ch, ch);
    }

    return 0;
}
```

Q#04: Write a program using while loop that ask the user to input any number and display that number in reverse order.

Answer:

```
#include <stdio.h>


int main() {
    int num, reversed = 0, remainder;
```

```
    // Taking input
    printf("Enter a number: ");
    scanf("%d", &num);

    // Reversing the number
    while(num != 0) {
        remainder = num % 10;
        reversed = reversed * 10 + remainder;
        num /= 10;
    }

    // Displaying the reversed number
    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

Q#05: Write a program using do while loop that print the sum of first 10 decimal digits.

Answer:

```
#include <stdio.h>
```

```
int main() {
    int sum = 0, i = 0;

    // Loop to calculate sum of first 10 decimal digits
    do {
        sum += i;
        i++;
    } while(i < 10);

    // Displaying result
    printf("Sum of first 10 decimal digits: %d\n", sum);

    return 0;
}
```

Name: Mumtaz Ali
Roll No: 24AI-36